Theory days in Rõuge, 28.01.2007

Framework for Fast Prototyping of Secure Computations

Dan Bogdanov
University of Tartu
db@ut.ee

Joint work with Sven Laur

What this talk is about

- Privacy concerns in data analysis
- Methods of preserving the privacy of data donors
- Overview of our privacy preserving data aggregation engine

What is considered private/sensitive?

If we made a survey and asked people about their

- health information
- political preferences
- sexual behaviour

then the answers will be considered **sensitive data**.

Problem statement

From the privacy requirements comes a problem:

- No participant except for the donor should see the answers.
- We still want statistical results on the data...
- The data miner must analyse the data without seeing ”too much” of it.

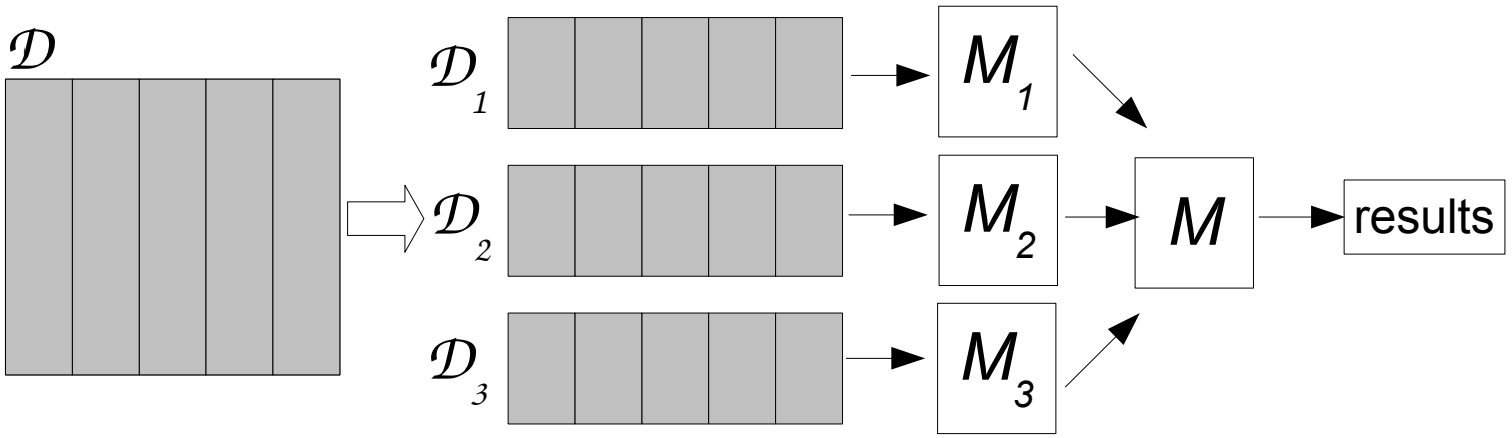
Theoretical setup

- Let $W_1, \ldots, W_n$ be the participants who provide us with the data.
- Each participant answers m questions
- The database $\mathcal{D}$ is a $n \times m$ matrix of answer values.

For the current example, let's say that each participant gives us one row of the database.

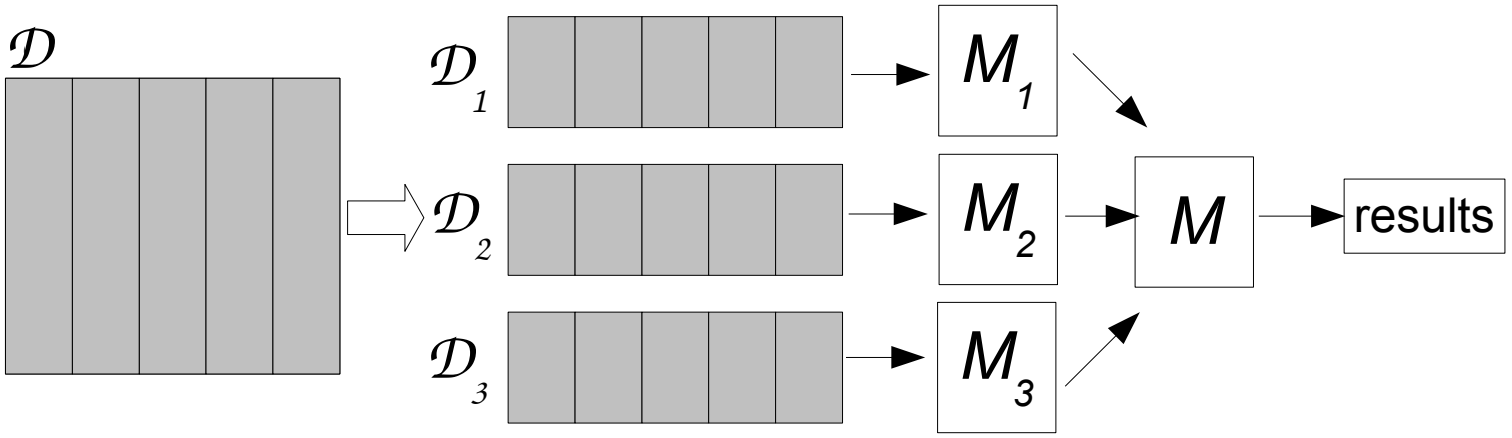
Idea 1: Distribute the rows

We have multiple data miners. Each one gains access to a subset of rows and calculates statistics on these rows.



In the example, $\mathcal{D}$ is the database. $\mathcal{D}_1, \mathcal{D}_2$ and $\mathcal{D}_3$ are subsets of the database. M_1, M_2 and M_3 are data miners working on the subsets and M is the master miner, who combines the results of the miners.

Idea 1: Distribute the rows - verdict

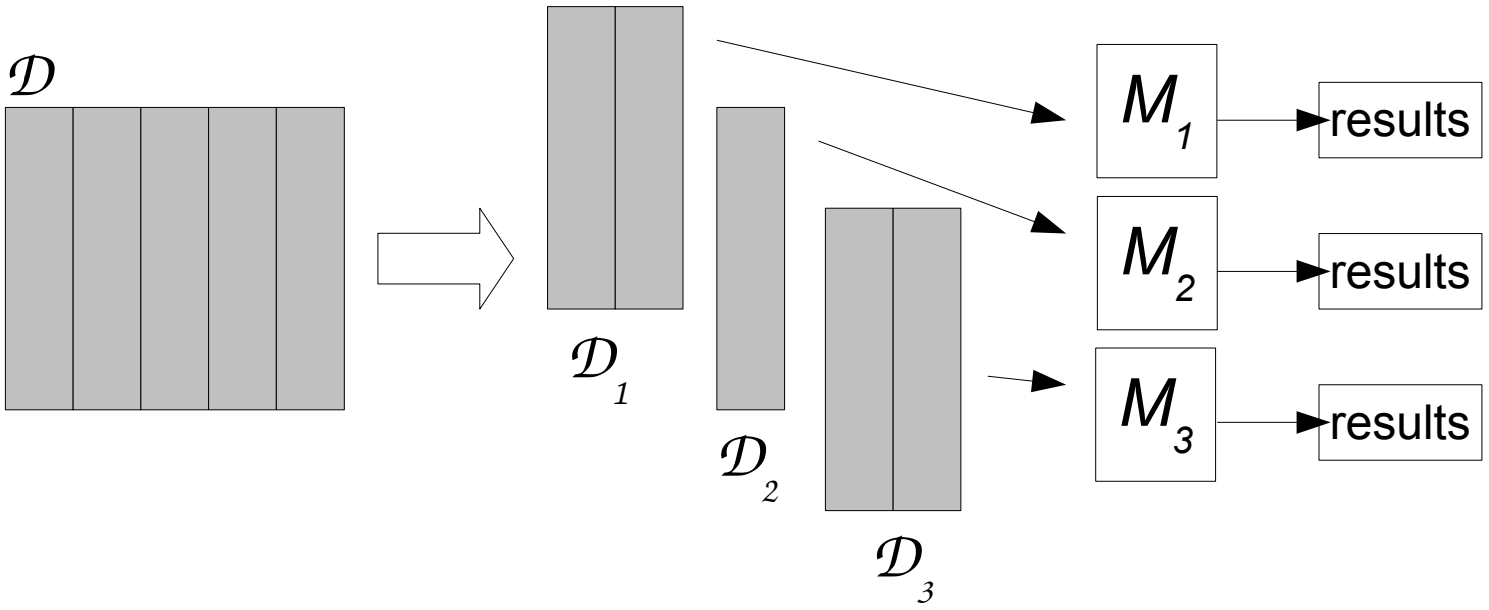


This method is not private - a miner can see complete information about a participant. Not every aggregation algorithm is easily distributable to requirements of this method

Method is not secure and therefore not acceptable.

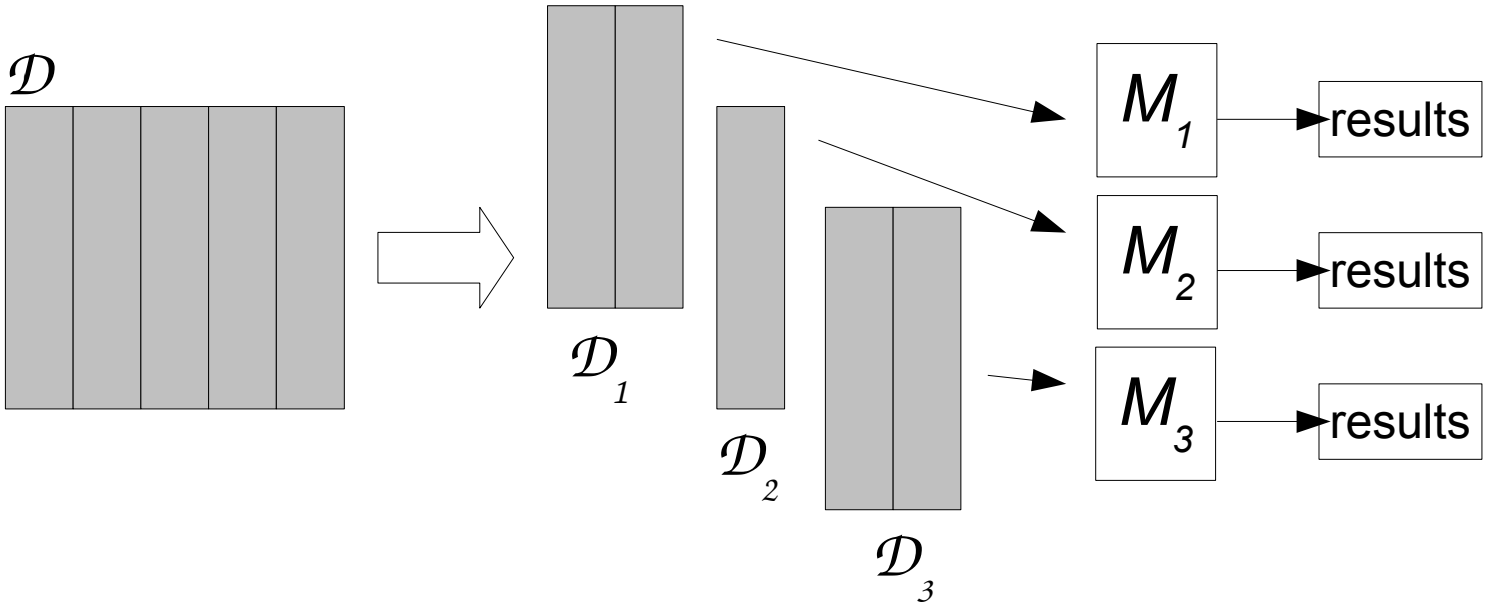
Idea 2: Distribute the columns

We have multiple data miners. Each one gains access to a subset of columns and calculates statistics based on these columns.



In this example $\mathcal{D}_1, \mathcal{D}_2$ and $\mathcal{D}_3$ are column subsets of $\mathcal{D}$. M_1, M_2 and M_3 are data miners working on the subsets.

Idea 2: Distribute the columns - verdict

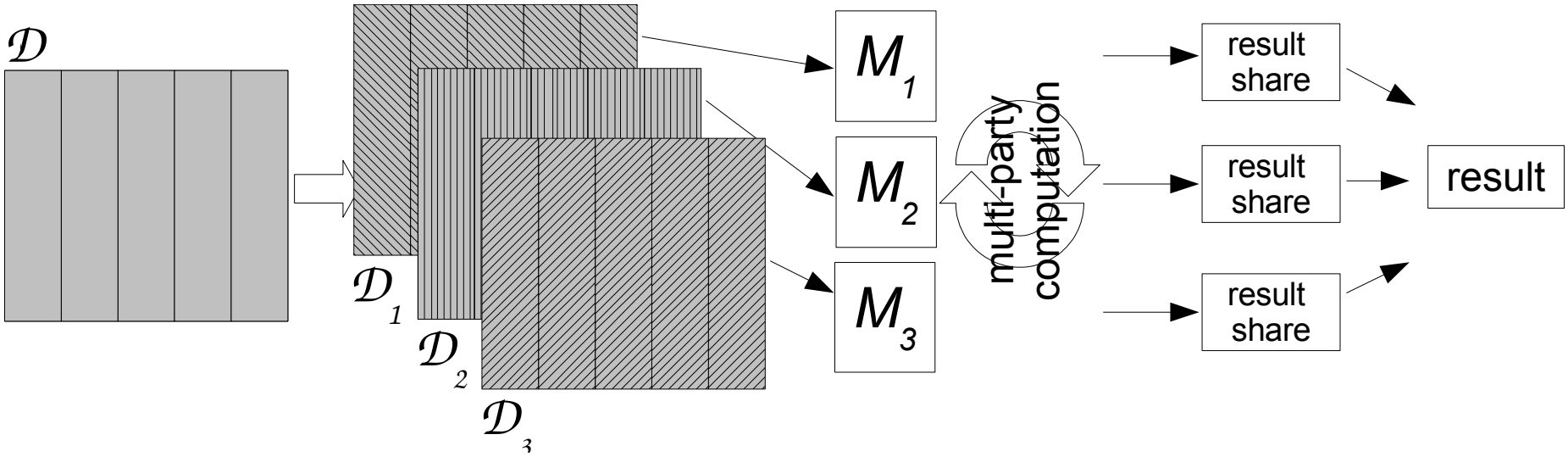


This method is not private - some miner may have enough attributes to access personal data. Since the miner can use only some attributes, its analysis capabilities are crippled.

Method is not secure and therefore not acceptable.

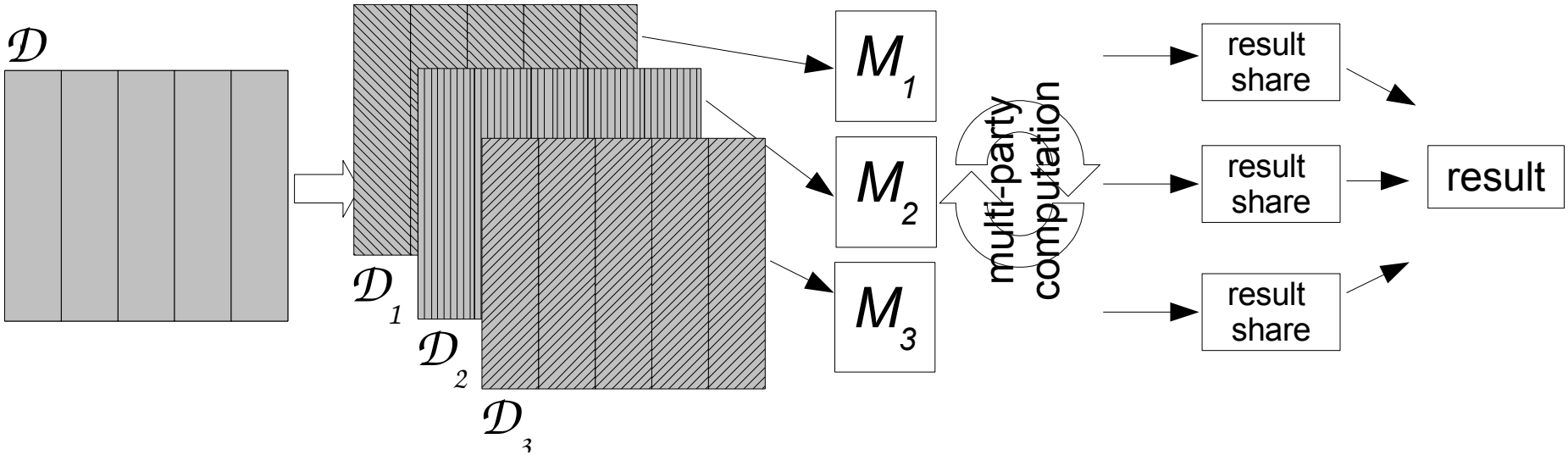
Idea 3: Distribute the values

We have multiple data miners. Each one is given a part of all the values. This is done by using **secret sharing**. The miners use **secure multi-party computation** to calculate aggregations.



$\mathcal{D}_1, \mathcal{D}_2$ and $\mathcal{D}_3$ contain shares of values in $\mathcal{D}$. M_1, M_2 and M_3 use multi-party calculation to calculate result shares.

Idea 3: Distribute the values - verdict



This method can be proven secure. However, most of the calculations and other operations require specific protocols, which can be time-consuming.

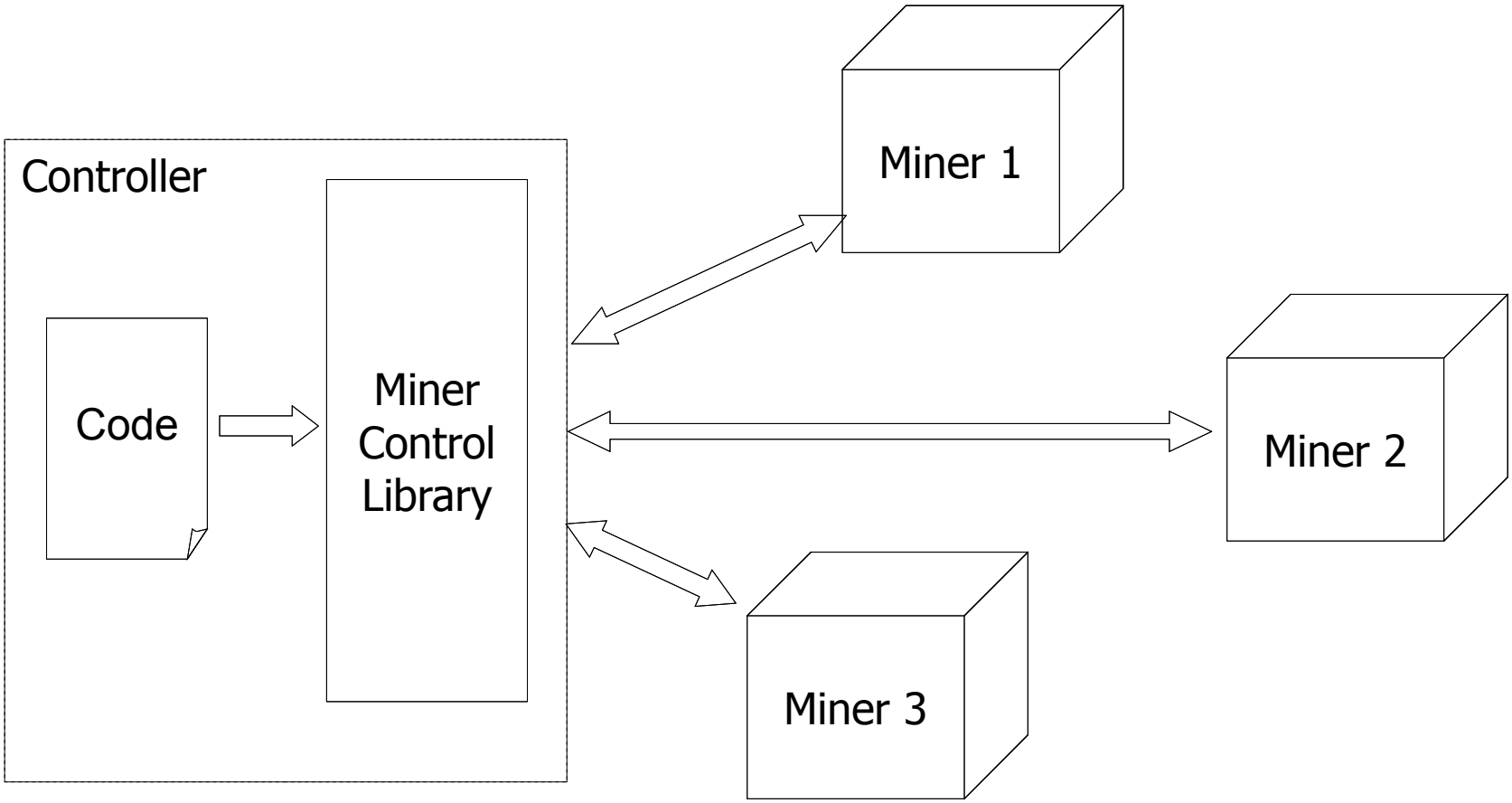
Method is secure, but is it feasible?

Breaking values into pieces

Secret sharing schemes in a nutshell:

- Assume that we have an input value s that we wish to keep secret.
- We have n nodes available for computation.
- We take s as the input and output n bitstrings $s_1, \ldots, s_n$ (shares).
- The value s can be reconstructed only if all shares are available.

Building the aggregation engine



Properties of the miners

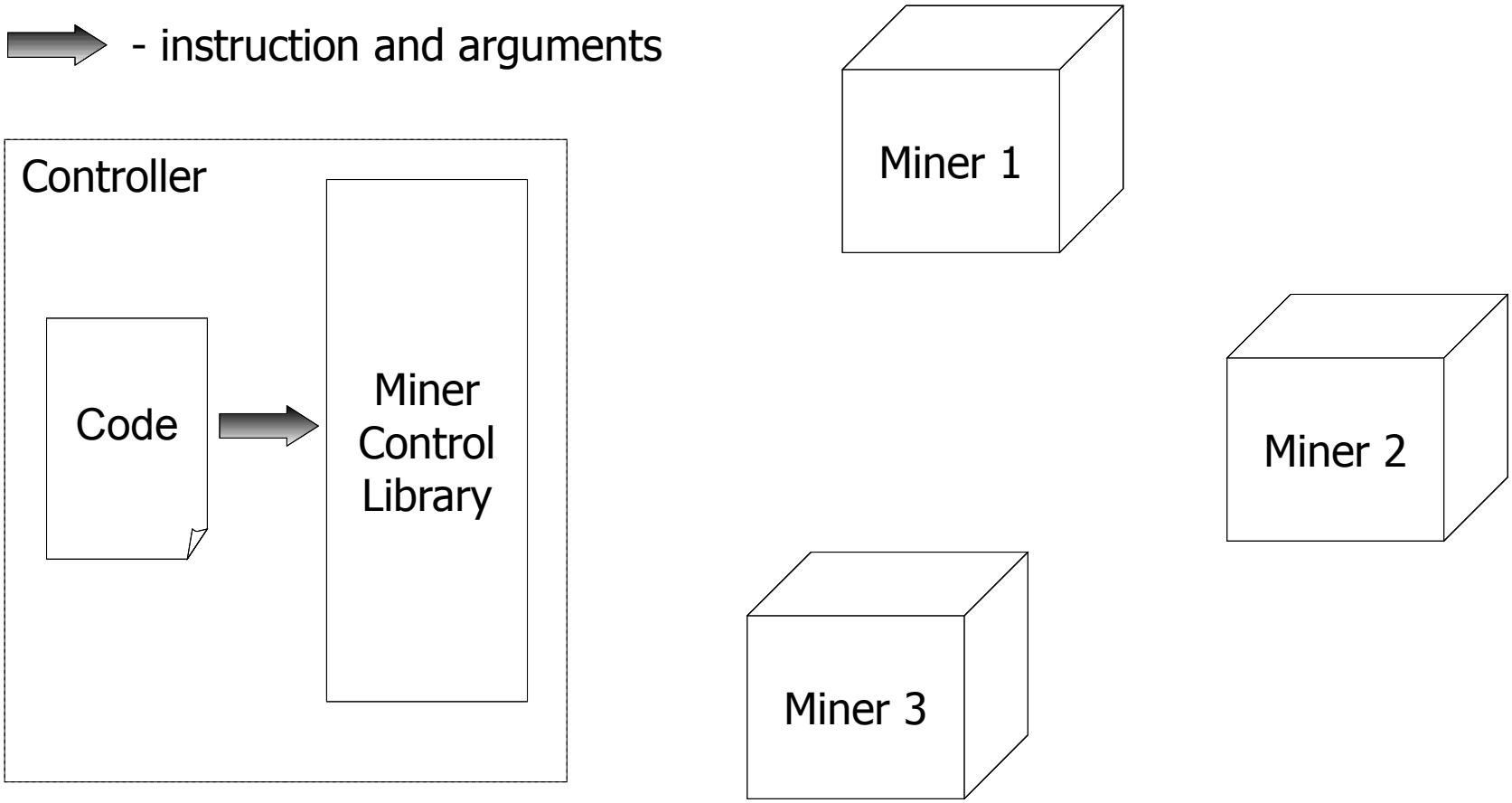
The distributed aggregation engine makes use of a number of facilities. Each miner has the following components:

1. Persistent storage — the database $\mathcal{D}$
2. Run-time storage — the heap $\mathcal{H}$ and the stack $\mathcal{S}$
3. Instruction scheduler for processing incoming commands
4. Network messaging for running protocols and exchanging data

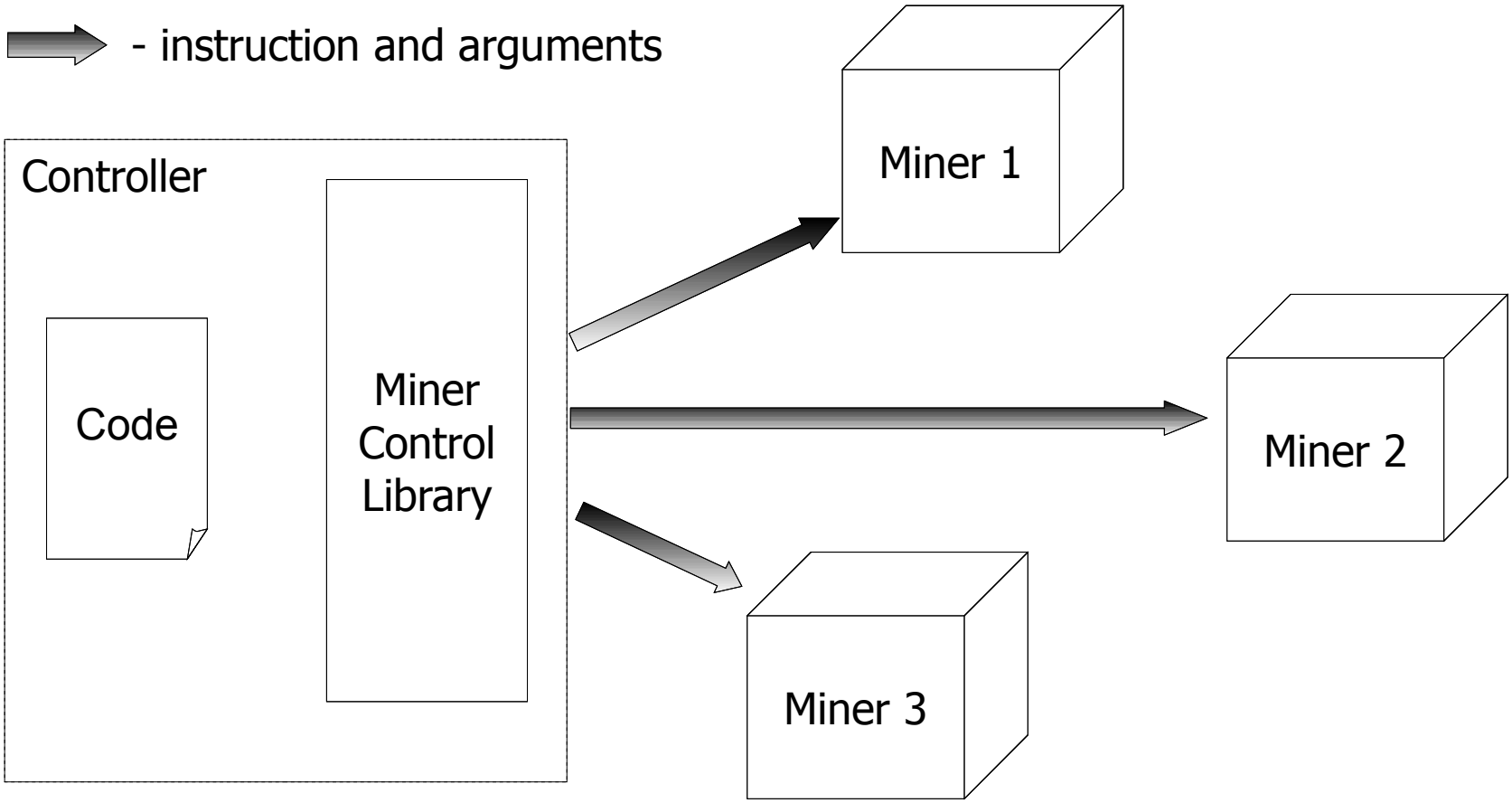
How secret sharing is used

- We need a homomorphic secret sharing scheme for our operations.
- Each miner stores only shares in its stack, heap and database.
- The controlling node does sharing and reconstructing.
- All operations will have to be synchronised on all nodes to make sure, that shares of the same value are used.

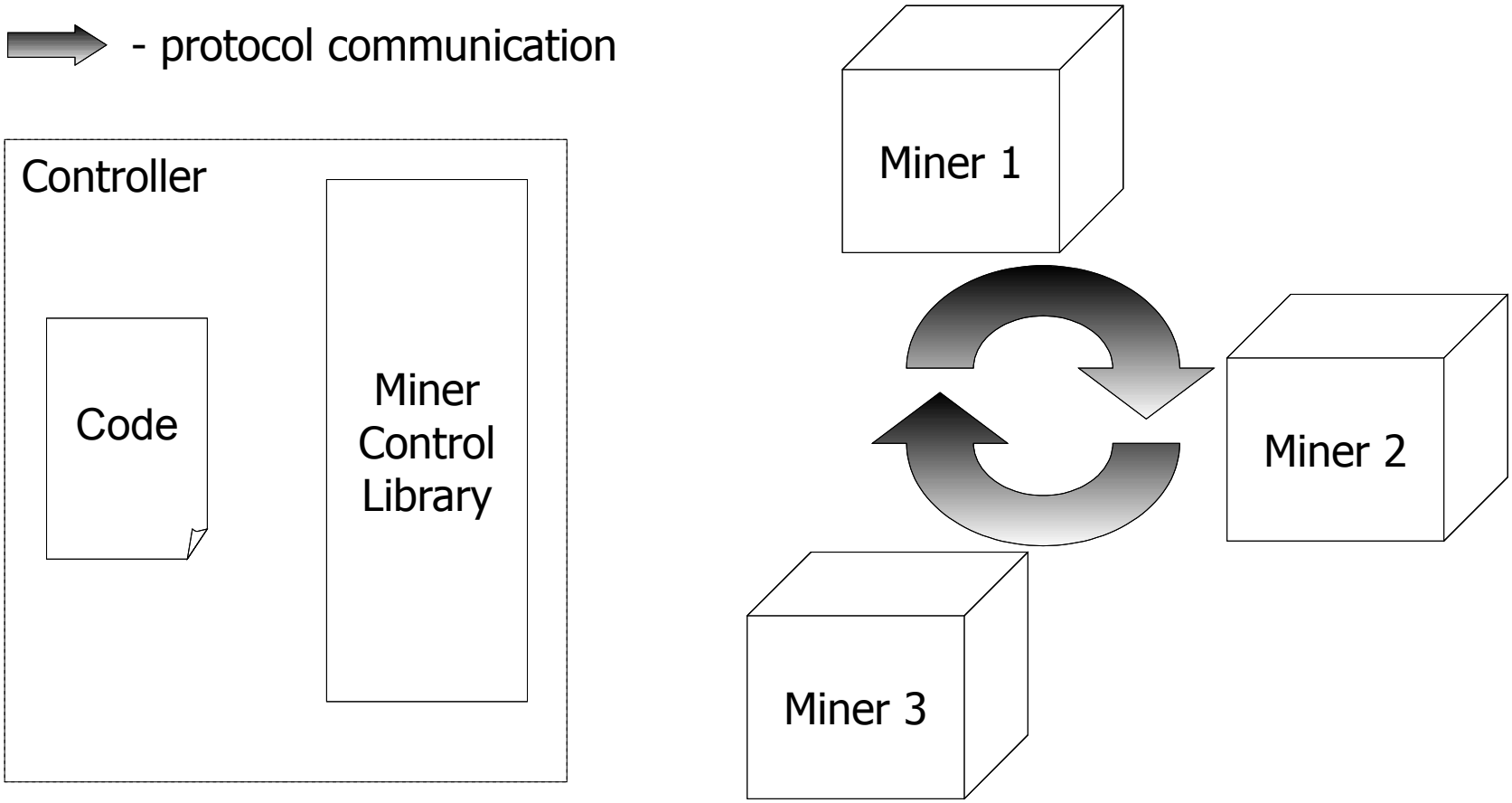
Instruction passing — 1



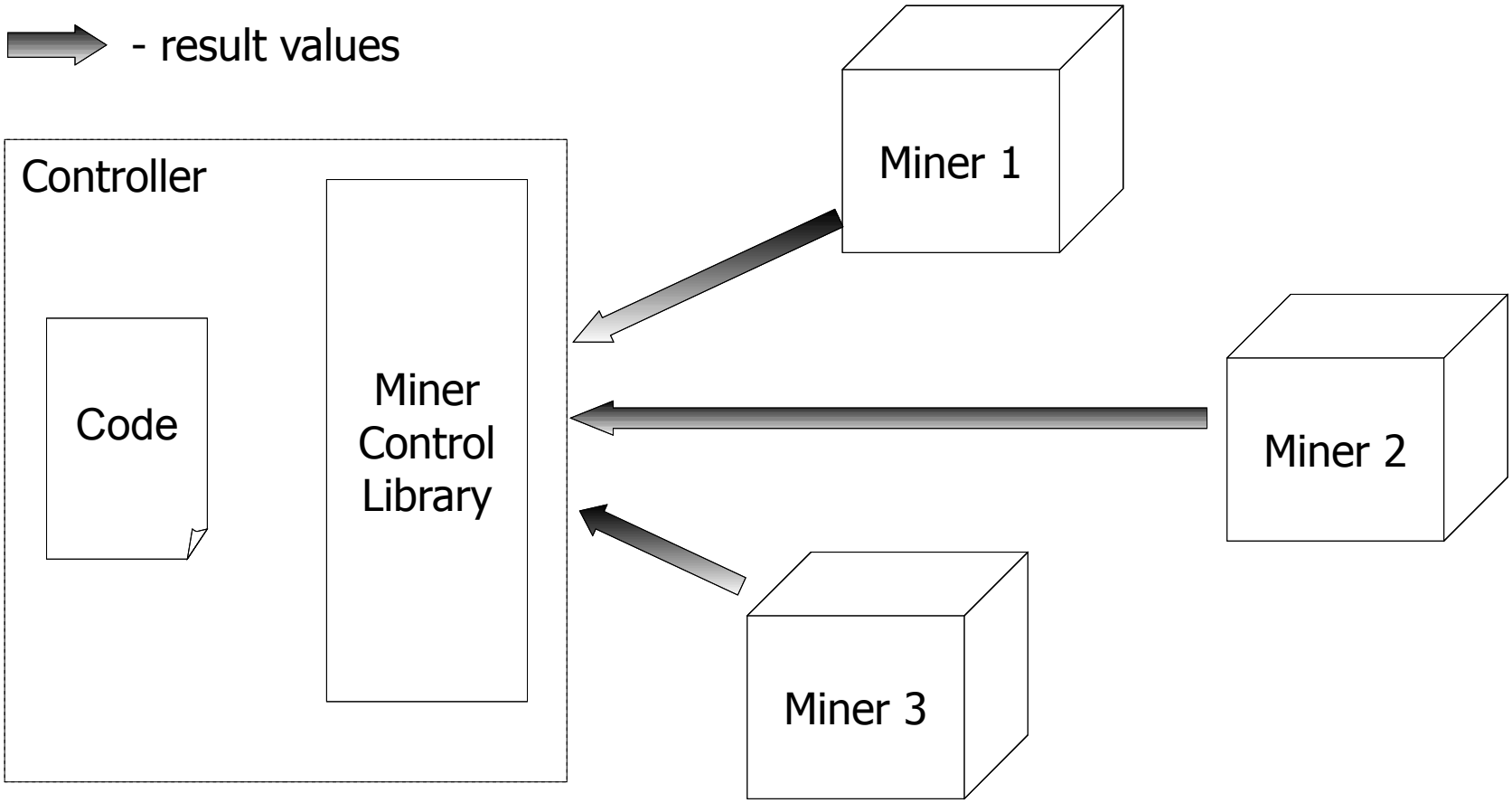
Instruction passing — 2



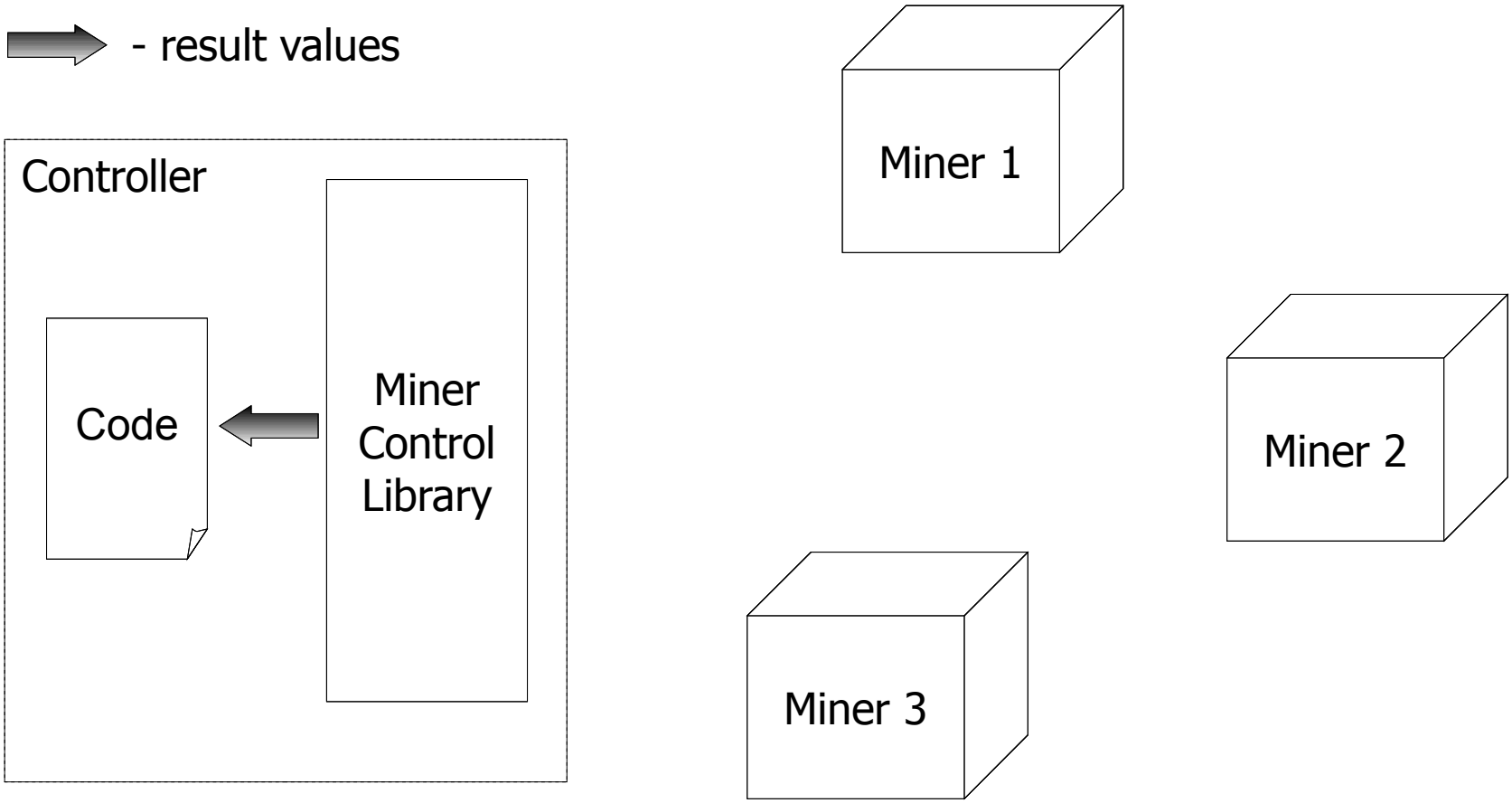
Instruction processing



Returning the result — 1



Returning the result — 2



Available operations

The engine can currently do:

- database, stack and heap manipulation
- adding and multiplication of value vectors
- conversion of shares in $\mathbf{Z}_2$ to shares in $\mathbf{Z}_{2^n}$
- greater-than predicate

Roadmap

The development is still work in progress.

- Currently we are developing the miner control library and the miner application.

- We want to use it to write a privacy-preserving implementation of:
  - ⋆ breadth-first search (APRIORI)
  - ⋆ depth-first search (FP-GROWTH)

End of talk

Thanks for listening!

Feel free to request demonstrations.